UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Desarrollo de un sensor de electrorrecepción para su implementación en robots submarinos

Cristian Catalin Tatu

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de un sensor de electrorrecepción para su implementación en robots submarinos

Autor: Cristian Catalin Tatu
Tutor: Carlos García Saura
Ponente: Francisco de Borja Rodríguez Ortiz

julio 2020

Cristian Catalin Tatu
Desarrollo de un sensor de electrorrecepción para su implementación en robots submarinos

Cristian Catalin Tatu

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Todo ocurre en la mente
y sólo lo que allí sucede
tiene una realidad.

George Orwell

Resumen

El Grupo de Neurocomputación Biológica estudia el sentido eléctrico que presentan algunas especies de peces, en concreto los del género Gnathonemus Petersii. Estos peces habitan aguas muy turbias y la evolución les ha dotado de un ”sexto sentido” de tipo eléctrico que complementa su visión y orientación, y su comunicación con otros peces. El órgano eléctrico de estos peces emite pequeños impulsos eléctricos de baja intensidad, a modo de ”radar biológico” que les permite detectar obstáculos y navegar en las turbias aguas de su hábitat.

La robótica submarina puede beneficiarse ampliamente de sistemas bioinspirados en el sentido eléctrico de los peces. Este proyecto aborda el desarrollo de un sensor de electrorrecepción para su implementación en robots submarinos, con el objetivo de poder detectar obstáculos en el entorno del robot.

Para que los sensores puedan detectar los obstáculos de forma eficaz hay que determinar donde se van colocar, qué tamaño van a tener y qué rango de corrientes usarán. El software desarrollado permite buscar automáticamente los valores de estos parámetros que generen una detección lo mas lineal posible. Además, se ha diseñado para ejecutarse en entornos de computación distribuida, logrando realizar más de 90 simulaciones concurrentes en el clúster de la EPS de forma eficiente.

En concreto, los resultados obtenidos usando Elmer para la simulación del campo eléctrico, han permitido identificar una configuración de electrodos optimizada para la detección de obstáculos de tipo pared a grandes distancias. Todo el software desarrollado se ha publicado como software libre.

Palabras clave

robot, submarino, campo eléctrico, electrorreceptor, simulación, pez eléctrico, software libre

Abstract

The Biological Neurocomputing Group studies the electrical sense of some fish species, specifically Gnathonemus Petersii. These fish inhabit dark waters, and evolution has endowed them with an electric ”sixth sense” that complements their vision, their spatial orientation, and their communication with other electric fish. The electrical organ of these fish emits small low-intensity electrical impulses, a sort of biological radar that allows them to detect obstacles and navigate the murky waters of their habitat.

Underwater robotics can benefit from bioinspiration on the electrical sense of these fish species. This project addresses the development of an electro-reception sensor for its implementation in underwater robots, with the aim of being able to detect obstacles within the robot's environment.

In order for the sensors to detect obstacles effectively, it is necessary to decide the placement, the size and the range of currents for the electrodes used. Our software allows to automatically search for the values of these parameters that generate the best linear-like detection curve possible. In addition, it has been designed to run in distributed computing environments, achieving more than 90 concurrent simulations in the faculty's cluster.

Specifically, the results obtained using Elmer for the simulation of the electric field have allowed us to identify an optimized electrode configuration for the detection of wall-type obstacles at high distances. We have published our resulting work as free software.

Keywords

robot, submarine, electric field, electroreceptor, simulation, electric fish, free software

Índice

Listas

Lista de códigos

Lista de figuras

Lista de tablas

1

Introducción

1.1. Estado del arte

El sentido eléctrico que poseen determinadas especies de peces [1,2] les proporciona la capacidad de emitir y percibir pequeños impulsos eléctricos en su entorno acuático (electrorrecepción). Esto les permite interactuar con otros peces eléctricos, y también percibir los obstáculos y características de su entorno más cercano. La electrorrecepción les da gran ventaja, especialmente en condiciones de baja visibilidad, donde hacen uso de los campos eléctricos para complementar sus otras modalidades sensoriales.

La electrorrecepción en robótica, inspirada en la biología, hace uso de corrientes eléctricas para detectar obstáculos debajo del agua [3]. Se han realizado múltiples estudios [4–6] y la electrorrecepción ha demostrado su efectividad en determinadas tareas autónomas [7–9]. Nos centramos principalmente en un proyecto que trabaja con robots submarinos diseñados y construidos usando impresoras 3D [10]. Estos robots constan de pequeños electrodos de acero inoxidable colocados sobre su superficie, que tienen el rol de electro-localizadores y permiten al robot submarino navegar por su entorno usando solamente campos eléctricos.

En la EPS UAM, Víctor Hugo Garcia desarrolló en su TFM [11] un prototipo de sensor capaz de detectar objetos debajo del agua captando la variación en el voltaje que produce tener un obstáculo conductor o aislante en el entorno del sensor. Este proyecto afronta la implementación de un sensor eléctrico a mayor escala en un robot comercial.

1.2. Motivación

Dentro del laboratorio GNB [1] en la EPS, recientemente dispusimos de un robot submarino para realizar más experimentos en entornos reales. Al escalar los resultados obtenidos previamente para implementarlos en el robot comercial, vimos que el rango de sensibilidad dependería mucho de la

[1] GNB: Grupo de Neurocomputación Biológica. `http://arantxa.ii.uam.es/~gnb/`

configuración de los electrodos. Además nos encontramos con la dificultad de realizar frecuentemente pruebas con el robot, dado que el entorno necesario es mucho mas grande que una pecera y tenemos que desplazarnos para realizar los experimentos.

Como ejemplo ilustrativo de la técnica de electrorrecepción que implementaremos tenemos la figura 1.1. En este experimento introdujimos dos cables recubiertos con plata (los electrodos) en una pecera y separados `10 cm` con el fin de medir la conductividad o resistencia eléctrica creada por el agua. Los cables fueron sumergidos `3 cm` y con la ayuda de un multímetro registramos el valor de la resistencia entre los dos electrodos. El valor de resistencia medido varía según la proximidad de

Figura 1.1: Fotografía de la medición de la resistividad del agua usando los electrodos recubiertos de plata.

los electrodos a la pared de la pecera y otros obstáculos. En concreto, la conductividad disminuye al acercar obstáculos aislantes. Esto es debido a que el agua actúa como una gran resistencia eléctrica que envuelve los electrodos en las tres dimensiones. Las corrientes eléctricas inducidas por los electrodos se ven afectadas por los obstáculos cercanos y esto es justamente lo que queremos utilizar para estimar la proximidad.

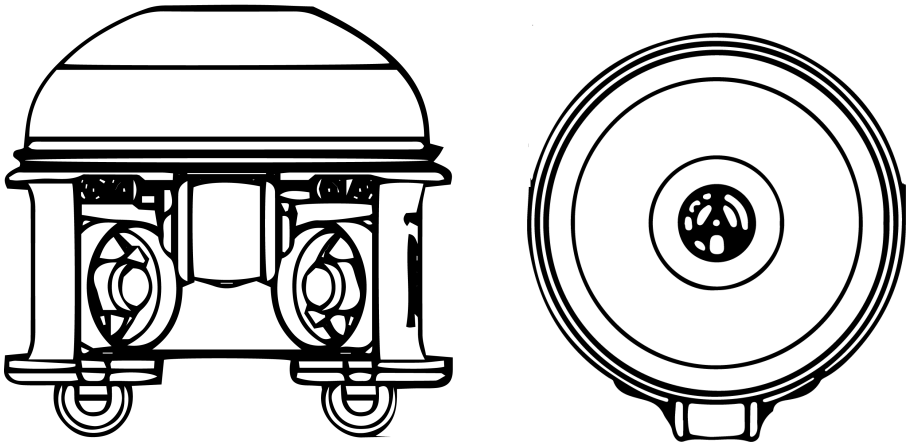
1.3. Robot submarino utilizado

Para este trabajo hemos modelado el Robot submarino "Seadrone™ Inspector 2" [12] (ver Figura 1.2). Este robot submarino está equipado con 5 motores para poder moverse en las 3 direcciones

y girar a los lados. Comercialmente se opera desde un iPad a través de la aplicación del fabricante, contando además con una cámara y 4 luces LED. Está diseñado para inspecciones de estructuras subacuáticas.

Figura 1.2: Robot submarino.

En la página web oficial de Seadrone™ nos proporcionan dibujos técnicos de este modelo (ver Figura 1.3). Las dimensiones aproximadas son: 30x30x30 cm.



(a) Vista alzado (b) Vista planta

Figura 1.3: Figuras técnicas del robot submarino. Fuente: seadronepro.com

Una imagen de la electrónica interior nos muestra los 5 controladores para los motores y una placa RaspberryPi 3B (en verde) que es la encargada de recibir los comandos del operador a través de la app y comandar los motores acorde a sus órdenes (ver Figura 1.4).

Este robot submarino ya tiene un método para la detección de obstáculos, la cámara. Sin embargo, en aguas turbias con poca visibilidad es inoperable. El pez eléctrico elefante, del que nos inspiramos, consigue navegar y evitar obstáculos en aguas con muy poca visibilidad usando el campo eléctrico que genera. Por tanto, podría ser muy beneficiosa una implementación parecida para nuestro robot usando electrorreceptrores.

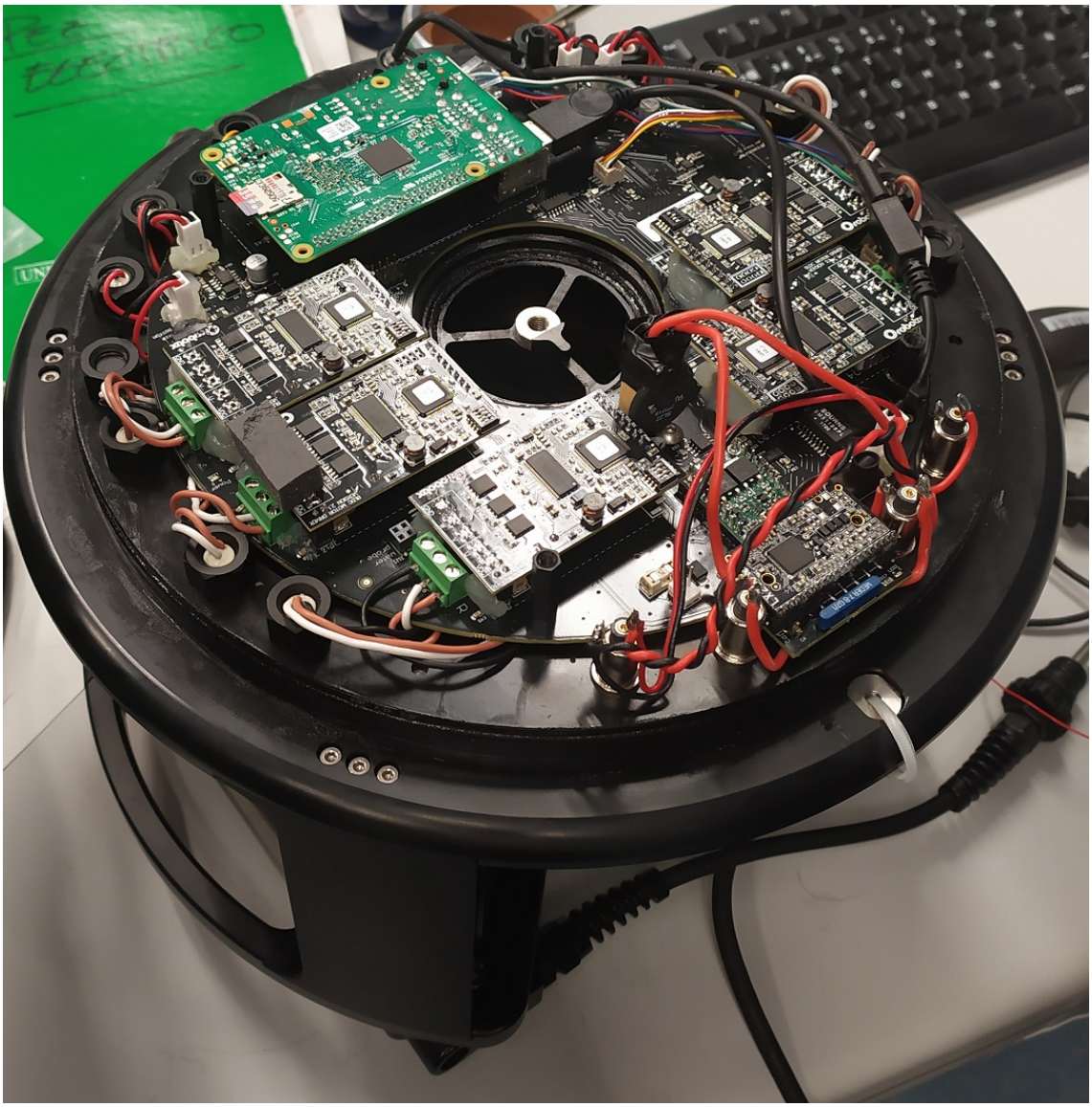


Figura 1.4: Electrónica interior del robot submarino.

1.4. Entorno de simulación para el robot submarino

El objetivo de este trabajo será determinar y parametrizar un flujo de trabajo para: a) crear un modelo virtual del robot submarino dentro de un ámbito subacuático y b) simular las interacciones eléctricas de sus sensores con los objetos del alrededor.

Este entorno constará de un modelo aproximado del robot (formado por semi-esferas y cilindros), y de un prisma que contendrá al robot y que representará el volumen del agua en el que estará sumergido. Además, el robot tendrá unidos a su superficie un par de semiesferas que representarán los electro-localizadores.

Para cada simulación, los comandos geométricos para el modelo 3D del robot serán creados desde un script de control de Python y transferidos a Gmsh que se encargará de su discretización (malla).

La mayoría de estos simuladores realizan los cálculos resolviendo ecuaciones diferenciales parciales (Ver pagina 103 del documento [13]). El método que usan para resolver estas ecuaciones se denomina 'Método de los elementos finitos' (Finite Element Method) [14]. Usando este método, Elmer, realizará los cálculos del campo eléctrico sobre una malla, es decir, una geometría discretizada en elementos finitos.

Para definir los parámetros de cada simulación se utilizan las condiciones de contorno. Por ejemplo, para crear una corriente eléctrica en un electrocereceptor con forma de esfera, la cara o superficie de la esfera necesitaría tener una condición de contorno que especificase la densidad de corriente que se desea aplicar.

2

Metodología de trabajo

La metodología que se ha utilizado para estudiar y analizar los campos eléctricos sigue el siguiente esquema:

$$Parametrizacion \Rightarrow Mallaje \Rightarrow Resolucion\,del\,Campo\,Electrico \Rightarrow Visualizacion$$

A continuación, describiremos el software utilizado para desarrollar las simulaciones del campo eléctrico en el medio acuático. Como base se ha optado por soluciones de Software Libre + Open Source (FOSS).

2.1. Parametrización con Python y OpenCASCADE

El script de control [1] ha sido el software diseñado para este trabajo de investigación usando Python [2] [15] en combinación con la librería de modelado 3D *OpenCASCADE* [3]. Su finalidad es crear, a partir de un fichero de configuración, la geometría necesaria para la realización de la búsqueda perimétrica. Además, se encarga de automatizar el uso del software encargado del mallado y la simulación FEM para que finalmente recopile los resultados y presentarlos al usuario. Ver Figura 2.1.

2.2. Mallado 3D con Gmsh

Gmsh [16] es un generador de malla de geometría en 2D y 3D de código abierto con dos motores/kernels CAD incorporados. Su objetivo es proporcionar una herramienta de mallado rápida, ligera y fácil de usar con entrada paramétrica y capacidades de visualización avanzadas. Nosotros usaremos dos de los módulos que Gmsh ofrece: geometría y mallado. La especificación de cualquier entrada a

[1] https://github.com/GNB-UAM/3D-electroreception-simulator

[2] Version 3.7 de Python

[3] https://www.opencascade.com/

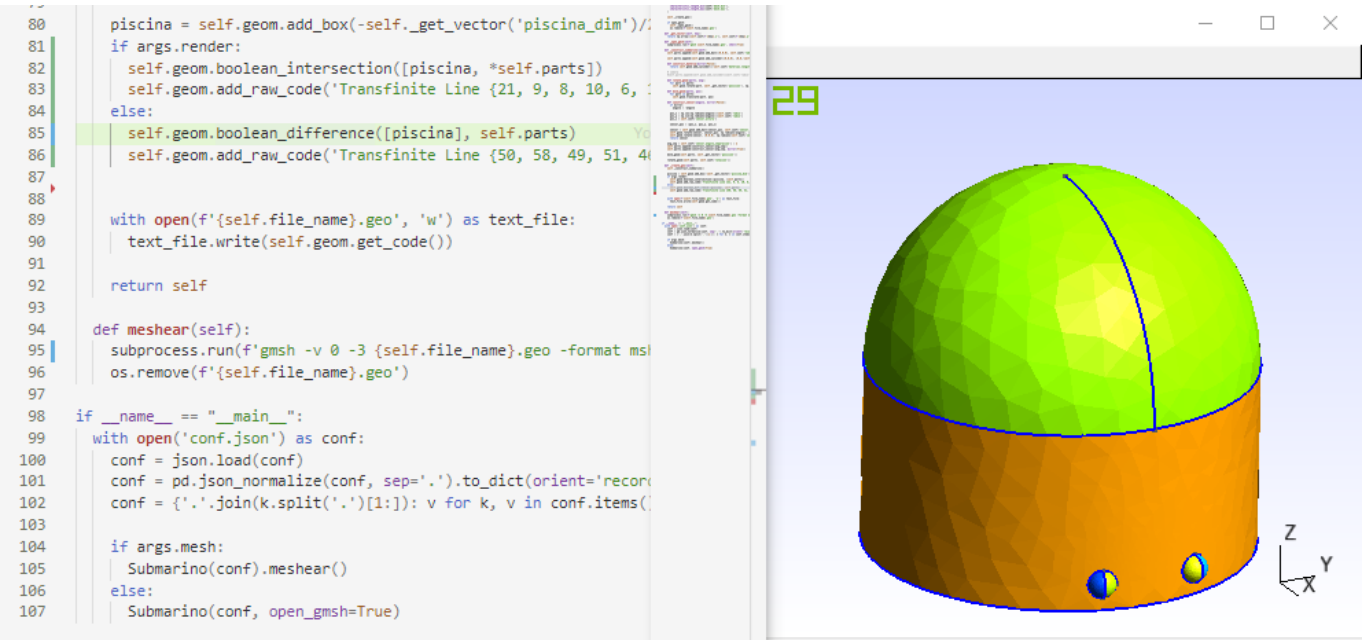


Figura 2.1: Modelo 3D generado con nuestro software mediante Python y OpenCASCADE.

estos módulos se puede realizar de forma interactiva utilizando la interfaz gráfica de usuario, o mediante archivos de texto ASCII utilizando el propio lenguaje de scripting de Gmsh (archivos `.geo`). Ver Figura 2.2.

Usaremos principalmente los archivos `.geo` para describir nuestra geometría (el submarino y el cubo que lo contendrá). Gmsh los interpretará para generar el mallado correspondiente. Cabe destacar que estos archivos se escribirán automáticamente y de forma paramétrica mediante nuestro software anteriormente mencionado. El mallado generado por Gmsh tiene tamaño variable, es decir, usará celdas de tamaño mas pequeño en zonas donde se necesita mas resolución y viceversa. Gmsh determina eso comprobando la relación de tamaño que tienen los objetos, si un objeto es muy pequeño en comparación otro, usará tamaños pequeños de celdas para poder definir bien el objeto de tamaño inferior. La versión de Gmsh que usaremos será la `4.1.5`.

2.3. Simulación con Elmer FEM

Elmer FEM [17] es un software de simulación multifísica de código abierto. Elmer incluye modelos físicos de dinámica de fluidos, mecánica estructural, electromagnética, transferencia de calor y acústica entre otros. Estos se describen mediante ecuaciones diferenciales parciales que Elmer resuelve mediante el Método de elementos finitos (FEM) para la malla pedida y las condiciones de contorno establecidas. Ver Figura 2.3.

Dentro de todos los módulos y ecuaciones que ofrece este software de simulación usaremos el mo-

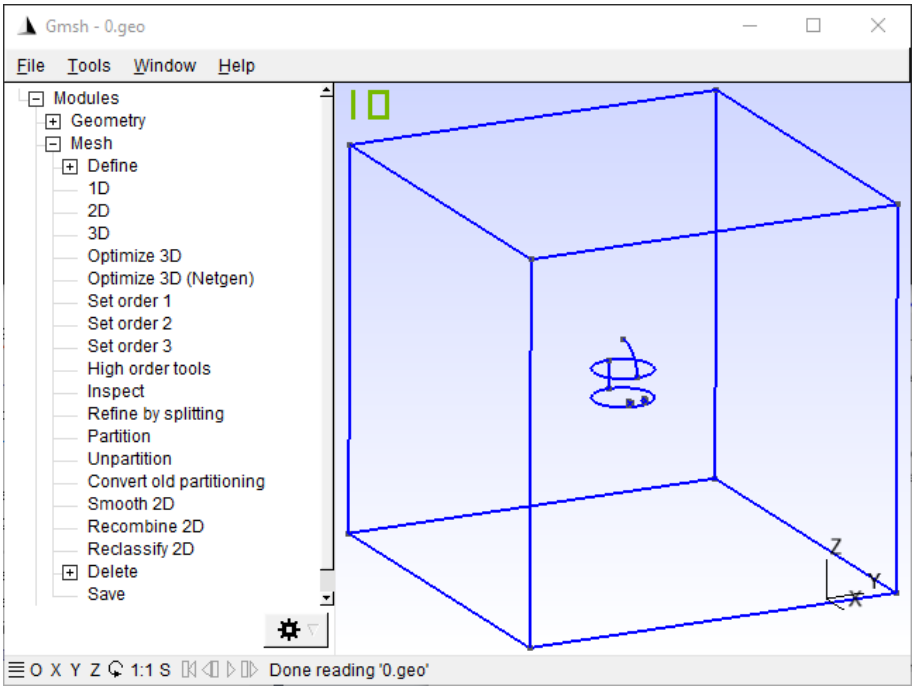


Figura 2.2: Captura de pantalla de la interfaz gráfica de Gmsh con la geometría CAD del robot submarino.

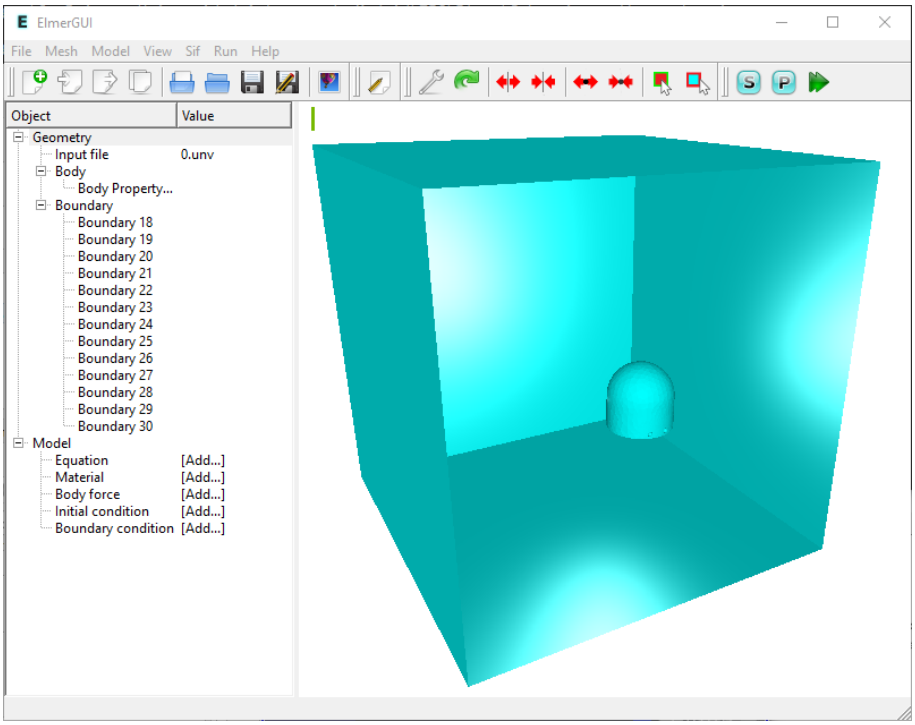


Figura 2.3: Captura de pantalla de la interfaz gráfica de Elmer con la malla del robot submarino.

dulo eléctrico ya que proporciona todas las condiciones de contorno necesarias (densidad de corriente y potencial eléctrico) para llevar a cabo las simulaciones. La versión de Elmer que usaremos será `8.4`.

2.4. Visualización con Paraview

Paraview [18] es una aplicación de visualización y análisis de datos multiplataforma de código abierto. Los usuarios de Paraview pueden rápidamente crear visualizaciones para analizar sus datos utilizando técnicas cualitativas y cuantitativas. La exploración de los datos se puede hacer interactivamente en 3D, o también mediante programación utilizando las capacidades de procesamiento por lotes de Paraview. Ver Figura 2.4.

El uso de este software es completamente opcional para el flujo de trabajo que estamos presentando, pero la visualización ayuda a validar que todos los parámetros y condiciones de contorno se han aplicado adecuadamente. La versión usada es `5.8.0`.

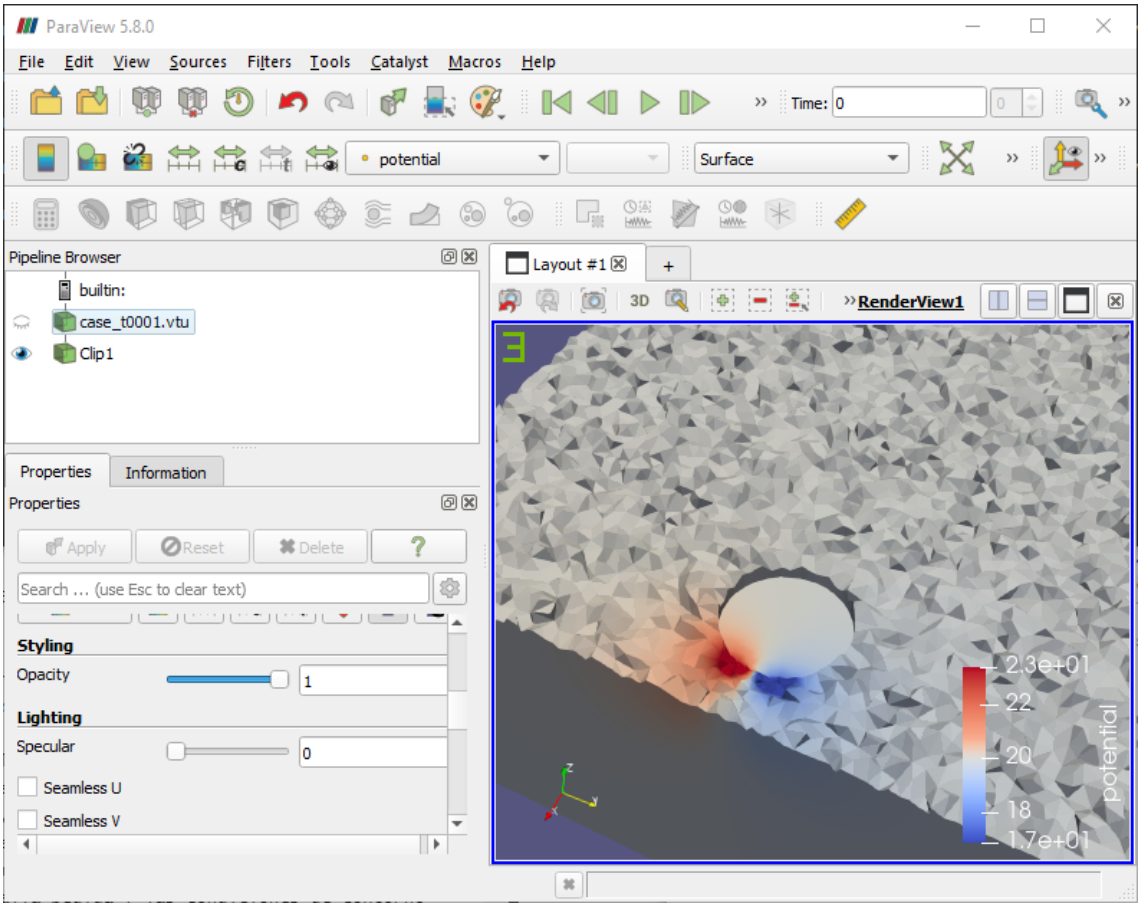


Figura 2.4: Captura de pantalla de la interfaz gráfica de Paraview.

3

Desarrollo del sistema de simulación

3.1. Procedimiento de la simulación del sistema

El procedimiento para la búsqueda de la posición óptima para los electroreceptores que llevará el robot sigue el siguiente flujo de trabajo.

Comienza con la ejecución de nuestro script de Python recibiendo un fichero JSON con la estructura de ejecución presentada en el Código 3.1. Todos los valores usan las unidades del sistema internacional. El fichero de configuración consta de dos partes fundamentales:

- La primera, con la clave `submarino`, denota la configuración que conformará la geometría y el mallado. En esa parte se especifican, entre otras cosas, las medidas que tendrá el robot (el radio y altura de su cuerpo) y también las dimensiones de los electro-receptores, así como el tamaño del volumen de agua en el que está sumergido, etc. También se especifican la posición y rotación absoluta del submarino dentro del volumen de agua. Esto nos será de gran ayuda ya que nuestro objetivo es optimizar el diseño de los electro-receptores para que tengan la sensibilidad adecuada a la hora de detectar un obstáculo.

- La segunda parte, con la clave `simulación`, fija las condiciones de contorno para el simulador Elmer. Para los experimentos hemos fijado una corriente de trabajo de `10mA` (aunque este valor se podría incrementar, por ejemplo en situaciones donde el robot no se aproxima a personas o animales). Para la conductividad del agua hemos elegido `0.041S/m` ya que es el valor utilizado en la literatura [19], y es un valor intermedio entre la conductividad del agua dulce y el agua marina.

Nuestro software de control está preparado para realizar las simulaciones de búsqueda paramétrica utilizando computación paralela en un cluster.

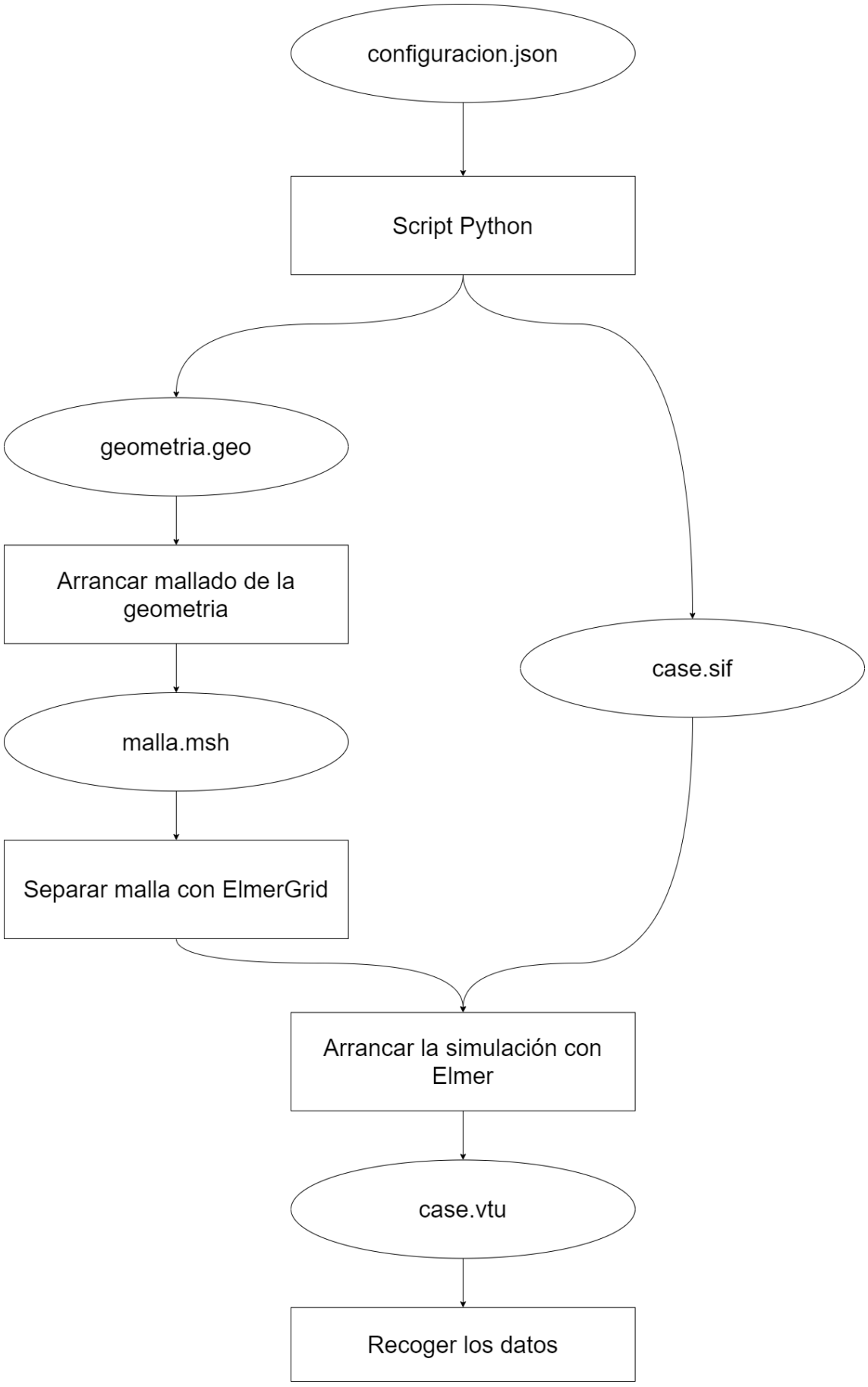


Figura 3.1: Procedimiento para resolver el campo eléctrico de una sola configuración.

Código 3.1: Ejemplo de la estructura del fichero de configuración para el script de python y sus posibles valores

```
{
    "submarino": { // Bloque de parametros del mallado
        "radio": 0.155, // Radio de la semi-esfera del robot submarino
        "altura_cuerpo": 0.147,
        "baterias": {
            "radio": 0.015,
            "longitud": 0.2
        },
        "sensor": {
            "tipo": "ESFERA",
            "radio": 0.01125, // Radio del electroreceptores en forma de esfera
            "angulo_separacion": 15,
            "altura": -0.125, // Posicion vertical sobre el robot de los sensores
            "rotacion": 0
        },
        "camara":{
            "radio": 0.035,
            "longitud": 0.071
        },
        "piscina_dim": { // Dimensiones del volumen de agua
            "x": 2,
            "y": 2,
            "z": 2
        },
        "posicion": { // Posicion absoluta del robot submarino
            "x": 0,
            "y": 0,
            "z": 0
        },
        "rotacion": 0, // Rotacion absoluta del robot submarino
        "mesh": {
            "min": 0.01, // Tamaño minimo de las celdas de la malla
            "max": 0.03,
            "refinamiento_mesh_esferas": 15
        }
    },


    "simulacion": {
        "corriente_sensor": 0.01, // Corriente en Amperios usada por los sensores
        "conductividad_agua": 0.041
    }
}
```

A continuación se muestran las 3 fundamentales librerías/módulos de Python usados:

1.– Pygmsh [20] cuyo principal objetivo es simplificar los comandos de `Gmsh` para poder generar geometría directamente desde código Python usando funciones. Este módulo proporciona primitivas para crear cubos, esferas, cilindros, etc. y además transformaciones geométricas.

2.– VTK [21], con el que conseguimos leer los ficheros de salida generados por Elmer al completar una simulación. Gracias a esta librería podemos extraer los valores de voltaje y corriente desde los resultados de cada simulación.

3.– MPI4PY [22] siendo totalmente opcional, acelera en gran medida la ejecución paralela de las simulaciones. Ofrece un puente entre Python y la biblioteca `Message Passing Interface` [23] (MPI) de computación paralela y distribuida. Será la que nos permita ejecutar las simulaciones en el clúster de la EPS.

Todos los módulos Python con sus versiones correspondientes se encuentran referenciados en el repositorio software del proyecto (Apéndice A).

Una vez que el script interpreta el fichero de configuración, genera el fichero `.geo` (ejemplo en Apéndice B) donde se define la geometría que tendrá el entorno (ver Figura 3.1). Esto es, el robot submarino estará formado por una semi-esfera, un cilindro que representa los motores y la batería del robot, y dos esferas que representarán los sensores (ver Figura 3.2). Cualquier parámetro de estos objetos se puede cambiar desde el fichero de configuración. Este archivo `.geo` se le pasa como input a Gmsh para generar el mallado de la geometría CAD.

El modelo CAD resultante está formado por un solo volumen, es decir un solo objeto constituye todo el entorno, el agua. Esto es, una vez creado el modelo del robot submarino y el volumen del agua, los diferentes objetos que componen al robot (el cuerpo y los sensores) se usan en una operación booleana de diferencia para dejar el volumen que ocupan hueco dentro del volumen de agua. Así simplificamos drásticamente la complejidad de la simulación y las condiciones de contorno se pueden aplicar directamente a las caras del (único) volumen del agua.

Paralelamente, el script crea el fichero de configuración de Elmer, que determinará el tipo de simulación que será llevada a cabo. Usando una plantilla del fichero de configuración de Elmer, `case.sif` (ejemplo en Apéndice B), que contiene las definiciones de entorno para estas simulaciones de campo eléctrico, el script rellena los campos necesarios con los valores especificados en el archivo original de configuración y lo guarda en una carpeta creada para esta simulación y que se limpia cuando se han recogido los resultados.

Cuando Gmsh haya finalizado la creación de la malla, todavía queda un paso antes de arrancar la simulación, y es convertir la malla a un formato preparado para Elmer. ElmerGrid hace exactamente eso, procesa un mesh y lo separa en diferentes componentes que Elmer pueda usar para ejecutar la simulación. Una vez acabada la simulación, Elmer guarda sus resultados en un fichero `.vtu` que el script de Python lee e interpreta con la ayuda del paquete VTK anteriormente mencionado. De ahí extrae el voltaje máximo, que será el voltaje medido en el sensor para el entorno y parámetros definidos.

El resultado de cada simulación se guarda en una estructura tipo diccionario junto a los parámetros que se usaron y los diferentes tiempos de ejecución. Finalmente se borran todas las simulaciones y los resultados se vuelcan sobre un fichero `json` que posteriormente podrá ser procesado usando cualquier herramienta de análisis de datos. Nosotros usaremos Jupyter Notebook y Pandas.

3.2. Barrido de parámetros

Código 3.2: Ejemplo de la configuración para un barrido de los parámetros del robot submarino

```
{
    "submarino": {
        "sensor": {
            "tipo": "ESFERA",
            "radio": ["lin", 0.0075, 0.015, 5],    // Definicion de un barrido de 5 pasos de tipo lineal
            "angulo_separacion": ["vals", 15, 45, 90], // Barrido de 3 valores directos
            "altura": -0.125,
            "rotacion": 0
        },
        "posicion": {
            "x": ["exp", 0, 1, 8],       // Barrido de 8 pasos de tipo exponencial
            "y": 0,
            "z": 0
        },
    },
}
```

Código 3.3: Ejemplo de la configuración para un barrido de los parámetros de la simulación

```
{
    "simulacion": {
        "corriente_sensor": ["exp", 0.01, 1, 5], // Defincion de un barrido de 5 pasos de tipo exponencial
        "conductividad_agua": 0.041
    }
}
```

Como nuestro objetivo es encontrar una configuración óptima de los parámetros del sensor que maximice la sensibilidad al detectar obstáculos, necesitamos la habilidad de probar de forma automática una gran variedad de parámetros. Para esto, en el fichero de configuración se puede indicar con una lista el tipo de barrido que se quiere hacer para cada parámetro, y el script de control generará la lista de valores que simular.

En el Código 3.2 tenemos un pequeño ejemplo de como indicar barridos del radio del sensor, el ángulo de separación entre los sensores y la posición del robot submarino dentro del agua, además de sus posibles tipos (líneas 5, 6 y 11). Los valores entre corchetes representan cada barrido. La sintaxis de los barridos es la siguiente:

```
["«tipo»", «valor_inicial», «valor_final», «numero_pasos»]
```

Nuestro script de control soporta 3 tipos principales de barridos:

1.– ***lin*** Crea una interpolación lineal entre el valor inicial y el final con el numero especificado de pasos. En la línea `5` de nuestro ejemplo (Código 3.2) el resultado sería la siguiente lista: `[0.0075, 0.0093, 0.0112, 0.0131, 0.015]`. Es el modo por defecto, si el primer argumento «`tipo`» se omite, se aplicará este modo.

2.– ***vals*** Este modo es especial, pensado para indicar directamente los valores que deben usarse, sin que se tenga que computar nada. Tomará literalmente los valores añadidos a la lista. En la linea `6` de nuestro ejemplo 3.2 el resultado sería la siguiente lista: `[15, 45, 90]`.

3.– ***exp*** Es parecido al primer modo, pero crea una interpolación exponencial en vez de una lineal. Es útil para parámetros donde se requiera una mayor precisión en un lado del rango que en otro. En la linea `11` de nuestro ejemplo (Código 3.2) el resultado sería la siguiente lista: `[0, 0.321, 0.637, 0.789, 0.824, 0.908, 0.961, 1]`. Se puede ver que hay 8 valores distribuidos exponencialmente.

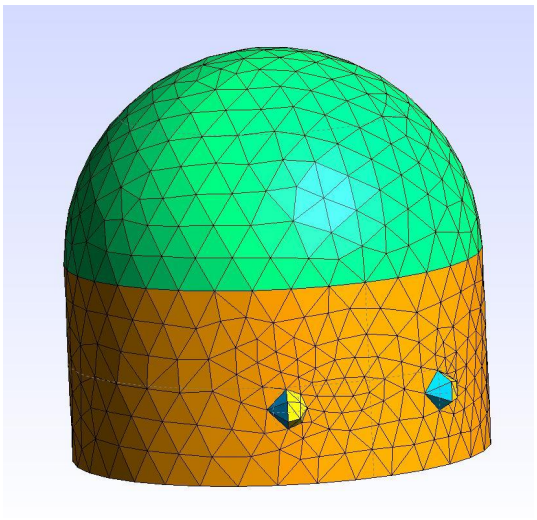
Para la configuración de la geometría del submarino, se pueden realizar barridos de múltiples parámetros como ejemplifica el Código 3.2. Adicionalmente, para los parámetros propios de la simulación (ver Código 3.3) los barridos múltiples tienen un menor coste computacional ya que no requieren crear una nueva malla en cada paso.

En la Figura 3.2 mostramos un ejemplo de las diferentes mallas que se generarían usando un barrido de parámetros. En este caso cambiamos el ángulo de separación, el radio y la altura a la que están colocados los sensores.

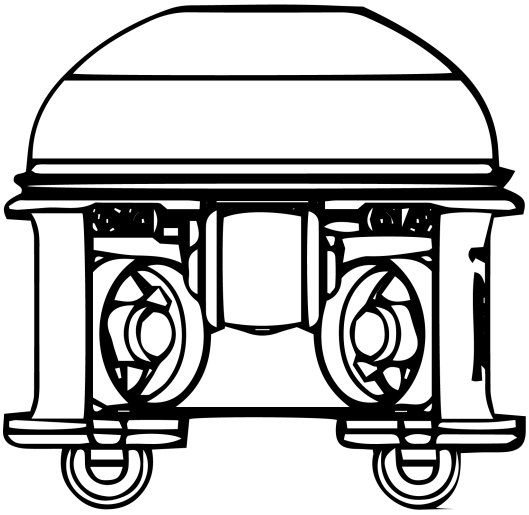
Finalmente, cuando el script de control haya interpretado y generado todas las listas de barridos, pasa a crear todas las posibles combinaciones de configuraciones de parámetros. Es decir, computará el producto cartesiano de todas las listas de parámetros consideradas como conjuntos, por tanto obtendrá N diferentes configuraciones a partir de:

$$N = \prod_{B} |b|$$

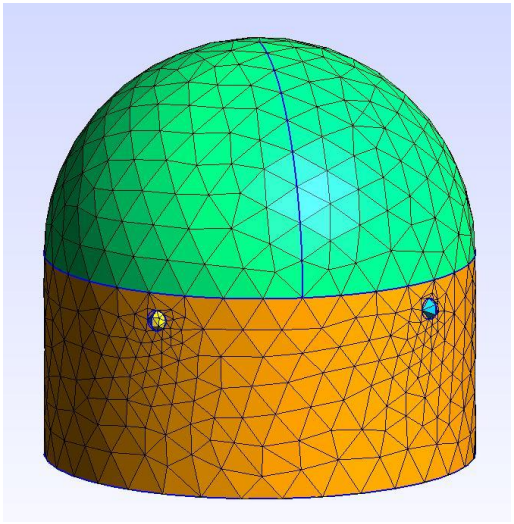
Donde B es el conjunto de los parámetros con listas de barridos y $|b|$ es la longitud o numero de pasos de cada lista. Por tanto, en el ejemplo anterior del Código 3.2, tendríamos $N = 5 \cdot 3 \cdot 8 = 120$ diferentes configuraciones, y por tanto, 120 diferentes mallas que computar y simular su campo eléctrico.



(a)



(b)



(c)

Figura 3.2: Dos figuras con mallas del robot submarino para diferentes configuraciones y un diagrama del robot real.

3.3. Ejecución de simulaciones

Cada configuración individual generada en el paso anterior determina una simulación concreta. Además, como cada simulación es completamente independiente de las demás, podemos aprovechar las capacidades de paralelismo que ofrecen las CPUs con varios núcleos para acelerar la tarea del barrido de parámetros.

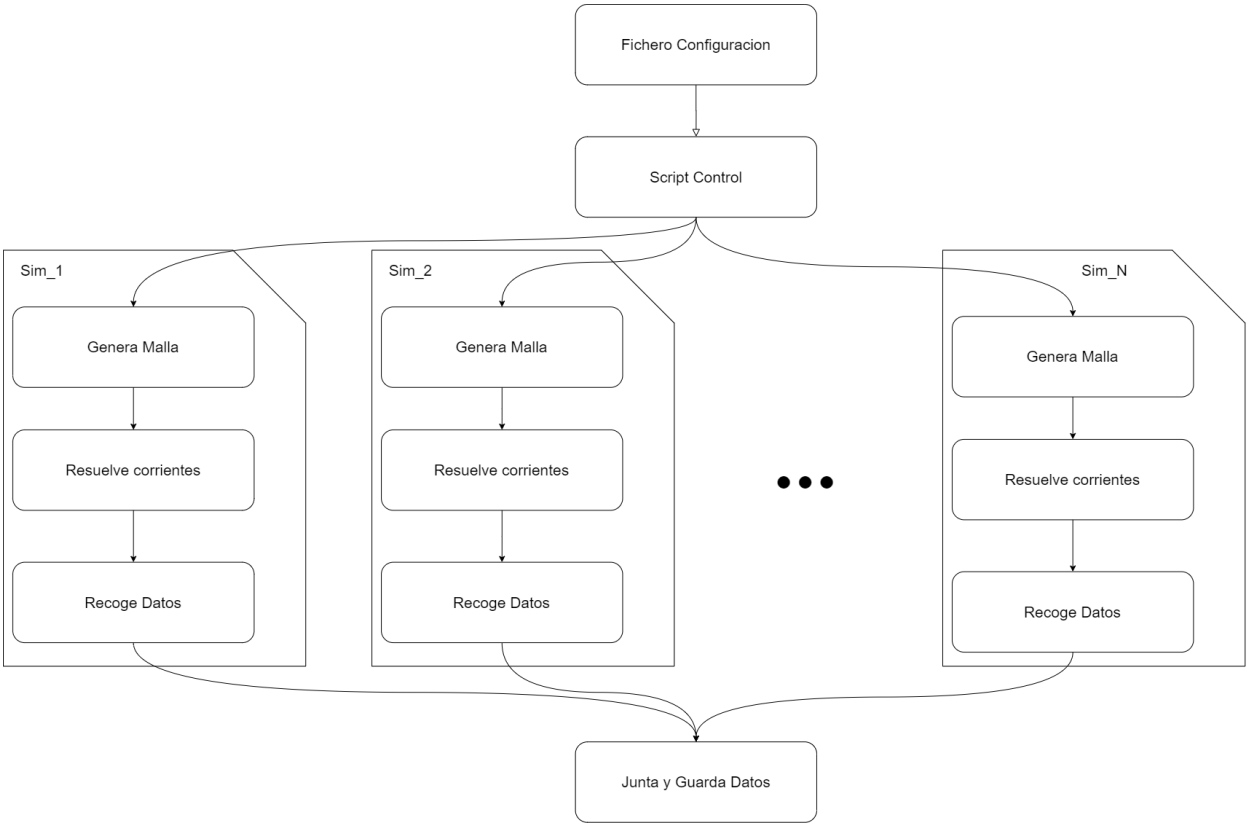


Figura 3.3: Diagrama de ejecución de las simulaciones.

Esto se visualiza en la Figura 3.3 donde cada simulación tiene respectivo flujo de trabajo, y al ser independientes, cada uno se puede ejecutar en un procesador diferente para aprovechar al máximo el paralelismo y reducir los tiempos de ejecución.

El script de control es el que lanza los comandos para mallar (`Gmsh`), resolver (`Elmer`) y recoger los datos (el propio script y opcionalmente `Paraview`). Al ejecutarse cada etapa por un proceso externo, se consigue paralelismo lanzando múltiples procesos para diferentes configuraciones, sin ser penalizados por el `GIL` [1] de Python.

[1] GIL: Global Interpreter Lock es un mecanismo de Python que prohíbe o bloquea la ejecución paralela de hilos en múltiples núcleos. Por defecto, todos los hilos creados en un proceso de Python se ejecutan en un mismo núcleo para evitar condiciones de carrera.

El clúster `labomat` de la EPS es un conjunto de 14 máquinas que comparten recursos para formar un sistema de computación mucho mas potente que cualquier máquina individual. Consta de 144 núcleos y más de 120 GB de memoria RAM. Para ejecutar las simulaciones nuestro script soporta 2 entornos de ejecución. Computación en una sola máquina y en múltiples máquinas (computación distribuida), que es el tipo de ejecución adecuada para el clúster `labomat`.

- Para la ejecución en una sola máquina lanzaremos un `Pool de hilos` [2], proporcionado por Python, con tantos hilos como núcleos disponibles haya y donde cada hilo se encargará de ejecutar las 3 etapas de cada simulación (ver Figura 3.3). El Pool de hilos de Python actúa de forma transparente para el usuario en cuanto a gestión de las tareas. A través del método *submit*, el usuario, en nuestro caso, manda ejecutar al Pool la función de simulación con una de las múltiples configuraciones como argumento. La función, que se encarga de crear la geometría, mallarla y simular el campo eléctrico con Elmer, devuelve el resultado del potencial simulado en cuanto termine y el pool lo recoge para el usuario.

- En un entorno distribuido hace falta un mecanismo para crear y administrar múltiples procesos en las diferentes máquinas que forman el clúster. Para esto usaremos `mpi4py` [22]. Una librería de Python introducida en el Capítulo 3.1. Con la ayuda de esta librería mantendremos la misma idea que en el punto anterior pero esta vez creando un Pool de procesos. Este se originará en un nodo llamado maestro y, usando la interfaz MPI, creará y gestionará múltiples procesos en las diferentes máquinas del clúster. En concreto usaremos como máximo 96 procesos (núcleos) para nuestras simulaciones.

Antes de poder ejecutar cualquier simulación en el clúster hace falta compilar e instalar todos el software necesario. En el apéndice A se detallan los pasos necesarios además de proporcionar un script de instalación automática.

[2] Pool de hilos: Es una estructura que gestiona un conjunto de hilos desde su inicio hasta su final. Además, para el programador es solo una unidad de procesamiento a la que se le mandan tareas en una cola y se ejecutan automáticamente en los diferentes hilos.

4

Resultados

4.1. Validación de las simulaciones

Elmer es un programa de simulación muy potente, pero su documentación es escasa y la interfaz de usuario es poco intuitiva. Por consiguiente, para confirmar la fiabilidad de los valores obtenidos con Elmer, vamos a plantear un problema de forma teórica para asegurarnos que los argumentos, las escalas y valores proporcionados son los correctos.

La geometría de esta simulación consta de 2 cilindros, uno encima de otro, con radio y altura `1 metro` (ver Figura 4.1).

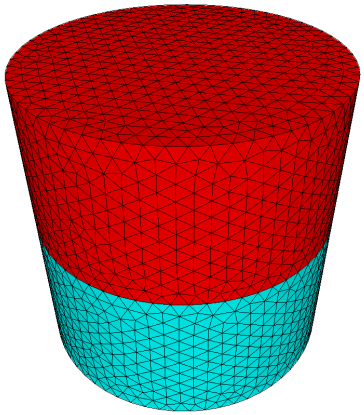
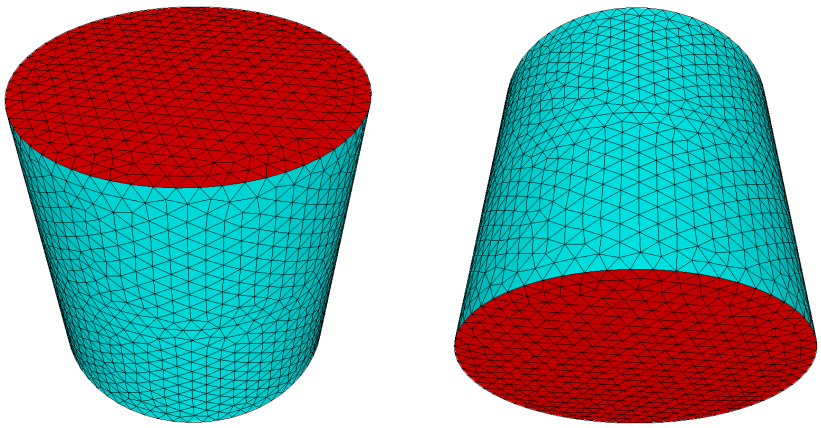


Figura 4.1: Vista de malla de geometría con dos cilindros apilados para la simulación de validación.

Cada uno tiene asignada una conductividad diferente, el cilindro superior (en rojo) tiene `1 S/m` y el inferior (en azul claro) `2 S/m`. Se han elegido estos valores para romper la simetría del problema ya que ahora la corriente encontrará mas resistencia en el cilindro superior y menos resistencia en el inferior.

El objetivo es hacer pasar una corriente de `1A` por la parte superior de los cilindros y una corriente de `-1A` en la parte inferior. De este modo, teniendo una conductividad diferente en cada cilindro, esperamos que en la cara superior se genere un potencial mayor en magnitud que en la parte superior (en concreto, el doble de tensión).



(a) Cara VCC con +1A (b) Cara GND con -1A

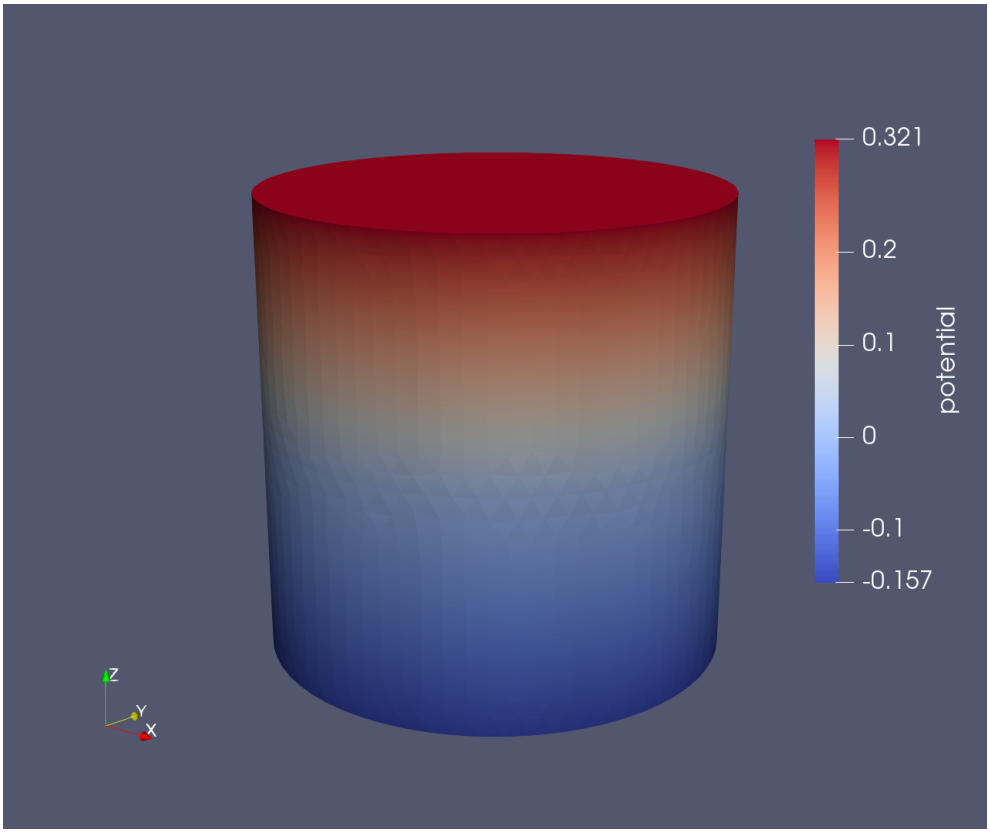
Figura 4.2: Las condiciones de contorno aplicadas a las caras de los cilindros.

Elmer solamente acepta asignar un valor de Densidad de Corriente a una o varias superficies al aplicar las condiciones de contorno. Por tanto, para aplicar 1 amperio a las dos caras del cilindro (de radio 1 m):

$$Densidad\,de\,Corriente\ = \frac{Corriente\,deseada}{Superficie\,cara} \Rightarrow\ \frac{1}{\pi} = 0{,}3183\ A/m^2$$

En la Figura 4.2 se resaltan en rojo las dos caras a las que se le ha aplicado una densidad de corriente de $0{,}3183\ A/m^2$ y $-0{,}3183\ A/m^2$ respectivamente.

En la Figura 4.3 que se respeta la relación esperada entre la diferencia de potencial que tienen las dos caras. La superficie del cilindro inferior tiene la mitad del valor del potencial en la cara superior y es de signo contrario ya que su densidad de corriente es negativa.



(a) Resultados de Elmer en Paraview

Figura 4.3: Diferencia de potencial generada por una corriente de 1 amperio en los dos cilindros apilados.

4.1.1. Simulación de experimento en una pecera

En la Figura 1.1 vimos una medición real con electrodos de cable plateado de 0.5 mm. En la Figura 4.4 hemos recreado en simulación el mismo experimento, mediante el procedimiento desarrollado (ver Capítulo 3.1). Esta simulación nos ha servido para identificar un problema que surge al usar este tipo de geometría (alambres muy finos en volúmenes grandes).

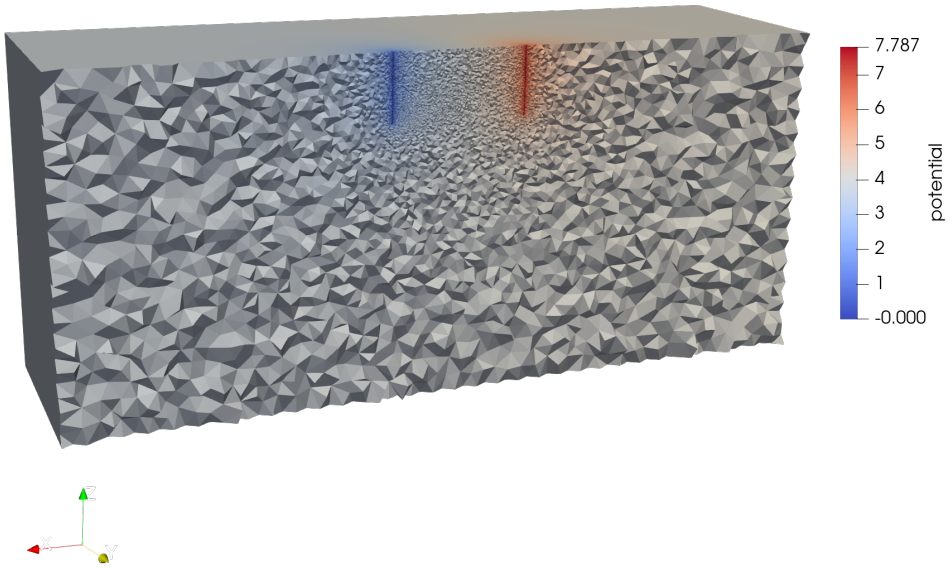


Figura 4.4: Potencial eléctrico distribuido en el volumen de agua de la pecera inspeccionado con Paraview.

Se puede ver el problema en la malla de la Figura 4.4. El volumen de la pecera está constituido por celdas que se reducen de tamaño cerca de los alambres. En ese punto la malla se tiene que hacer extremadamente fina para poder representar adecuadamente los cilindros de 0.5 mm de diámetro (llegando a producir ficheros de varios GB de RAM). Esta diferencia tan grande entre el tamaño mas grande de celdas y el mas pequeño hace que la simulación tenga resultados menos precisos, e incluso puede provocar inestabilidades que llevan al fallo de la simulación. Por tanto, decidimos utilizar electrodos esféricos ya que su geometría es mas simple y menos susceptible a estos errores. Esto hace que haga falta usar menos celdas, es decir mallas mas pequeñas, para conseguir el mismo nivel de precisión en la simulación.

4.2. Optimización de los electrorreceptores del robot

Finalmente, para la optimización de los electrodos usados en el submarino se han realizado 3750 simulaciones para barrer los siguientes parámetros:

- El radio de los electrodos esféricos
- El ángulo de separación de los electrodos
- La altura de los electrodos
- La posición del submarino con respecto a la pared

Para todos se probaron 5 valores diferentes, a excepción de la posición del submarino que se evaluó en 30 pasos para proporcionar mayor resolución en los resultados. Esto es necesario ya que al acercarse el submarino hacia la pared, la diferencia de voltaje requerida para pasar la corriente fijada de 10 mA crece exponencialmente. Por tanto, para el parámetro de la posición hemos usado el tipo `exp` al configurar el barrido y los demás parámetros los hemos dejado con la interpolación lineal por defecto.

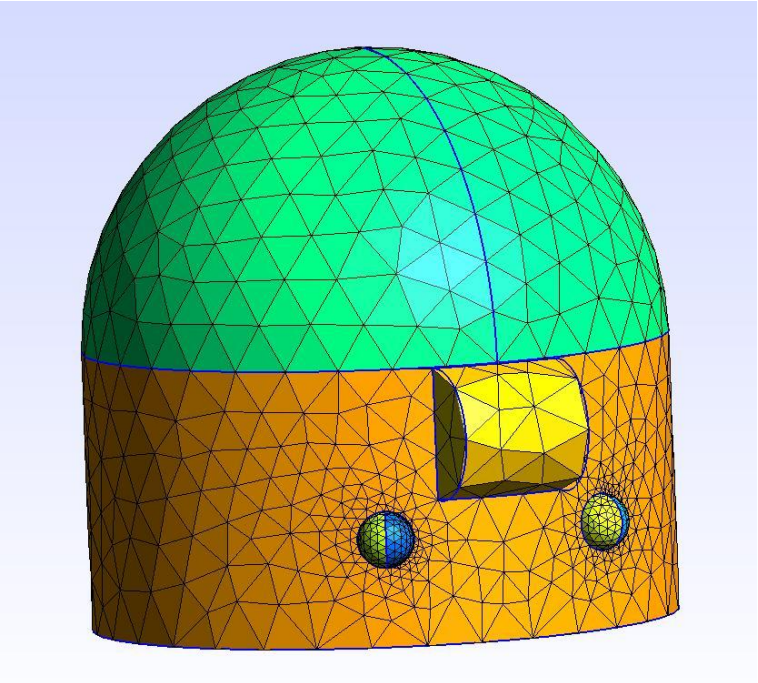


Figura 4.5: Imagen de la malla del submarino con los electrorreceptores.

Las 3750 simulaciones se realizaron en el cluster `labomat` y los resultados obtenidos contienen, para cada simulación, una entrada con los parámetros usados y la diferencia de potencial generada en los electrodos. Hemos usado nuevamente Python junto a sus librerías de graficación para visualizar como de sensibles sería cada configuración al detectar la pared de la piscina.

Agrupando todas las simulaciones por los 3 parámetros determinantes de los sensores obtenemos la Tabla 4.1. La tabla solamente tiene 125 entradas que se corresponden con los $5 \cdot 5 \cdot 5$ valores que se han barrido para estos parámetros. El total de 3750 simulaciones viene porque, para cada configuración distinta de parámetros, el submarino se fue acercando a la pared de la piscina (en 30 pasos) para registrar como cambia el potencial de los sensores.

Radio Sensores (mm)	**Ángulo Separación** (grados)	**Altura sensores** (cm)
7.5	15	-12.5
		-9.8
		-7.2
		-4.6
		-2
	31.25	-12.5
		-9.8
		-7.2
		-4.6
		-2
...	**...**	**...**
15	80	-12.5
		-9.8
		-7.2
		-4.6
		-2

Tabla 4.1: Tabla con los valores barridos de los 3 parámetros de los sensores. Solo se muestran los 5 primeros y últimos valores, la longitud de la tabla es de 125 entradas.

Las cotas de los parámetros, ambas incluidas, que se usaron para realizar los barridos han sido:

- Radio Sensores [7.5, 15] con tamaño de paso de 1.875 mm
- Ángulo Separación [15, 80] con tamaño de paso de 16.25 grados
- Altura sensores [-12.5, -2] con tamaño de paso de 2.62 cm

Cabe destacar que la altura de los sensores tiene valores negativos debido a las coordenadas absolutas de la geometría. La base de la semiesfera del submarino representa la altura 0 metros, y todo objeto ubicado por debajo tendrá coordenadas negativas.

En la Figura 4.6 elegimos una configuración aleatoria para dar un ejemplo de cómo cambia el potencial de los sensores al acercarse el submarino hacia la pared. El comportamiento exponencial que presenta el voltaje en la gráfica es debido a que, al moverse, queda cada vez menos agua entre el submarino y la pared aislante. En la zona cercana de los 10cm de distancia generamos una gran

densidad de valores que simular gracias al modo `exp` del fichero de configuración. Esto es para tener mejor resolución, ya que el potencial empieza a cambiar rápidamente en dicha región.

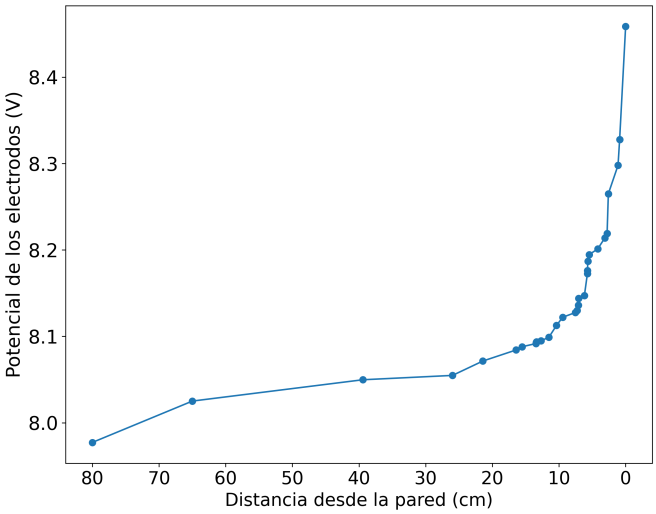


Figura 4.6: Potencial eléctrico de sensores con respecto a la distancia de la pared aislante.

Para nuestro objetivo de encontrar la combinación de parámetros más sensible, necesitamos analizar las curvas de voltaje de todas las 125 configuraciones. Consideramos que una configuración es sensible si la diferencia de potencial cambia notablemente cuando la distancia a un obstáculo disminuye. En nuestro caso, el obstáculo es la pared del volumen de agua. Para ello, primero vamos a visualizar cómo varían las distintas curvas superponiéndolas en una misma gráfica (ver Figura 4.7).

Como cada configuración tenia una distribución diferente del potencial, se han normalizado todas para poder compararlas en la misma escala. Vemos que la mayoría solamente presentan cambios sustanciales cuando el submarino está ya a 20 cm del obstáculo, sin embargo, hay una configuración que destaca sobre las demás y es la que más se aproxima a la curva de sensibilidad idónea.

La curva forma buscada para las curvas de potencial es la linea recta ya que es la que nos proporciona un cambio constante y fácil de predecir, ideal para los sensores de detecciones de obstáculos de rango elevado. Para determinar cual de las 125 configuraciones es la mas sensible, hemos elegido la curva que mas se parece a una linea recta $f(x) = x$ usando el método del error cuadrático medio que se define por:

$$ECM = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2$$

Donde, en nuestro caso, $\hat{Y}_i$ es la función $f(x) = x$ y Y_i es cada una de las 125 curvas. Computamos así el error de todas las configuraciones y elegimos la que minimiza este error. En la Figura 4.8

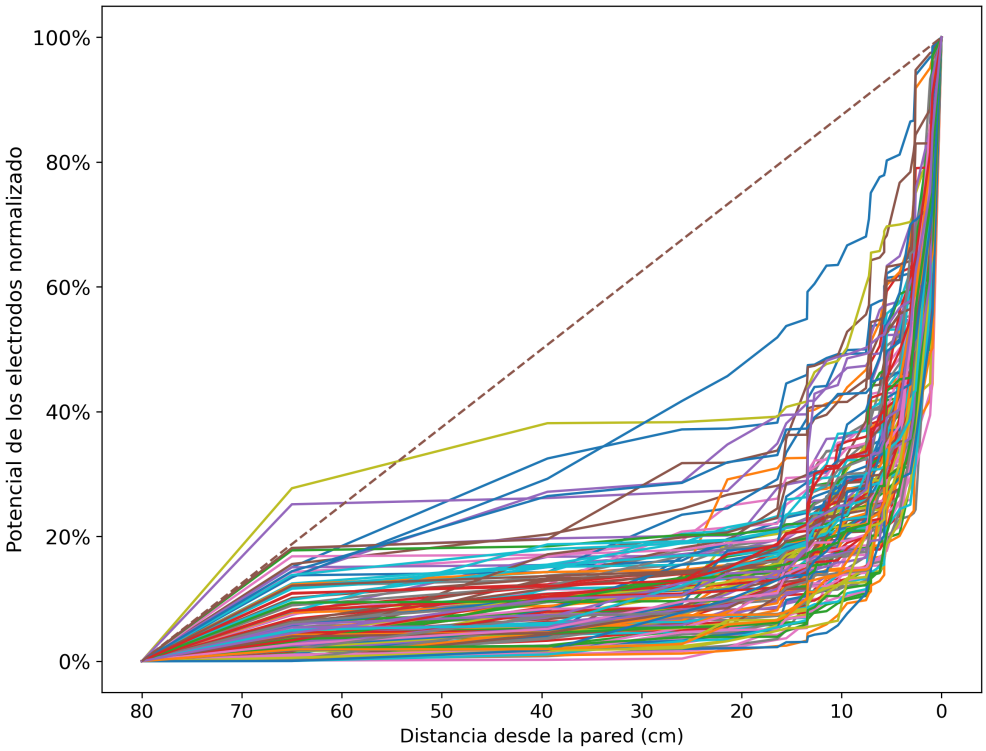


Figura 4.7: Superposición del potencial eléctrico de las 125 configuraciones al acercarse hacia la pared. El trazo punteado representa la respuesta lineal deseable.

mostramos las curvas más y menos sensibles según este criterio.

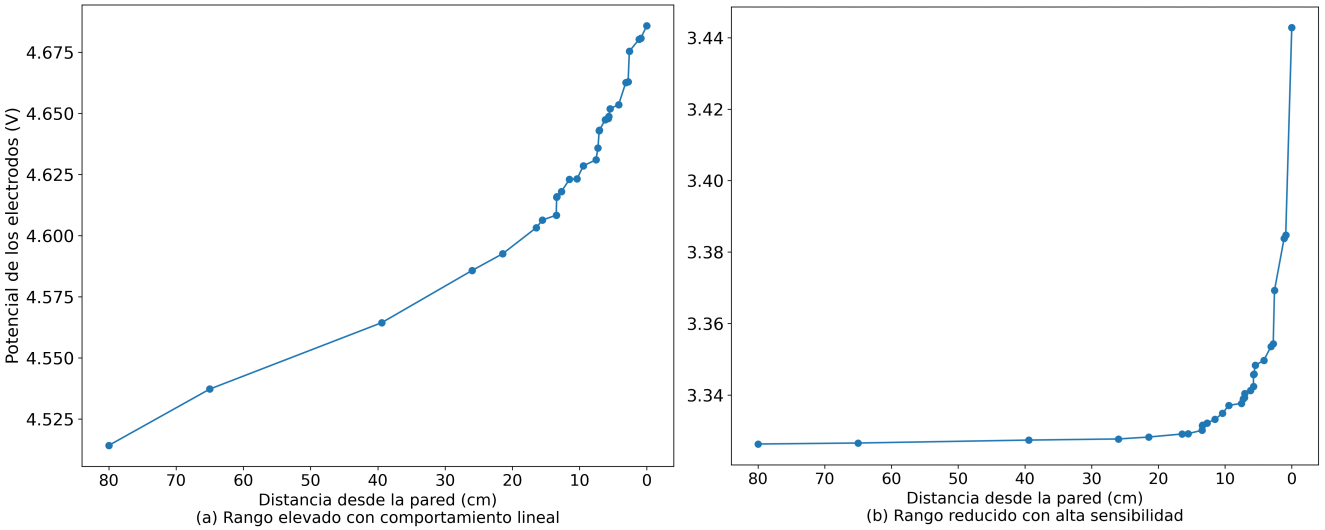


Figura 4.8: Potencial de la configuración de rango mas elevado con comportamiento lineal y la de menos rango con alta sensibilidad.

Se aprecia la diferencia entre la configuración con menos sensibilidad `(a)` y la que presenta la mejor sensibilidad `(b)`. La mas sensible empieza a tener cambios sustanciales desde distancias alejadas, mientras que los cambios de voltaje en la configuración menos sensible ocurren demasiado cerca del obstáculo.

La configuración de parámetros que ha producido la curva mas sensible la mostramos en la Tabla 4.2 junto a la malla final del submarino (Figura 4.9).

Radio Sensores (mm)	**Ángulo Separación** (grados)	**Altura sensores** (cm)
11.25	15	-12.5

Tabla 4.2: Tabla con los valores concretos que han producido la curva de detección mas sensible

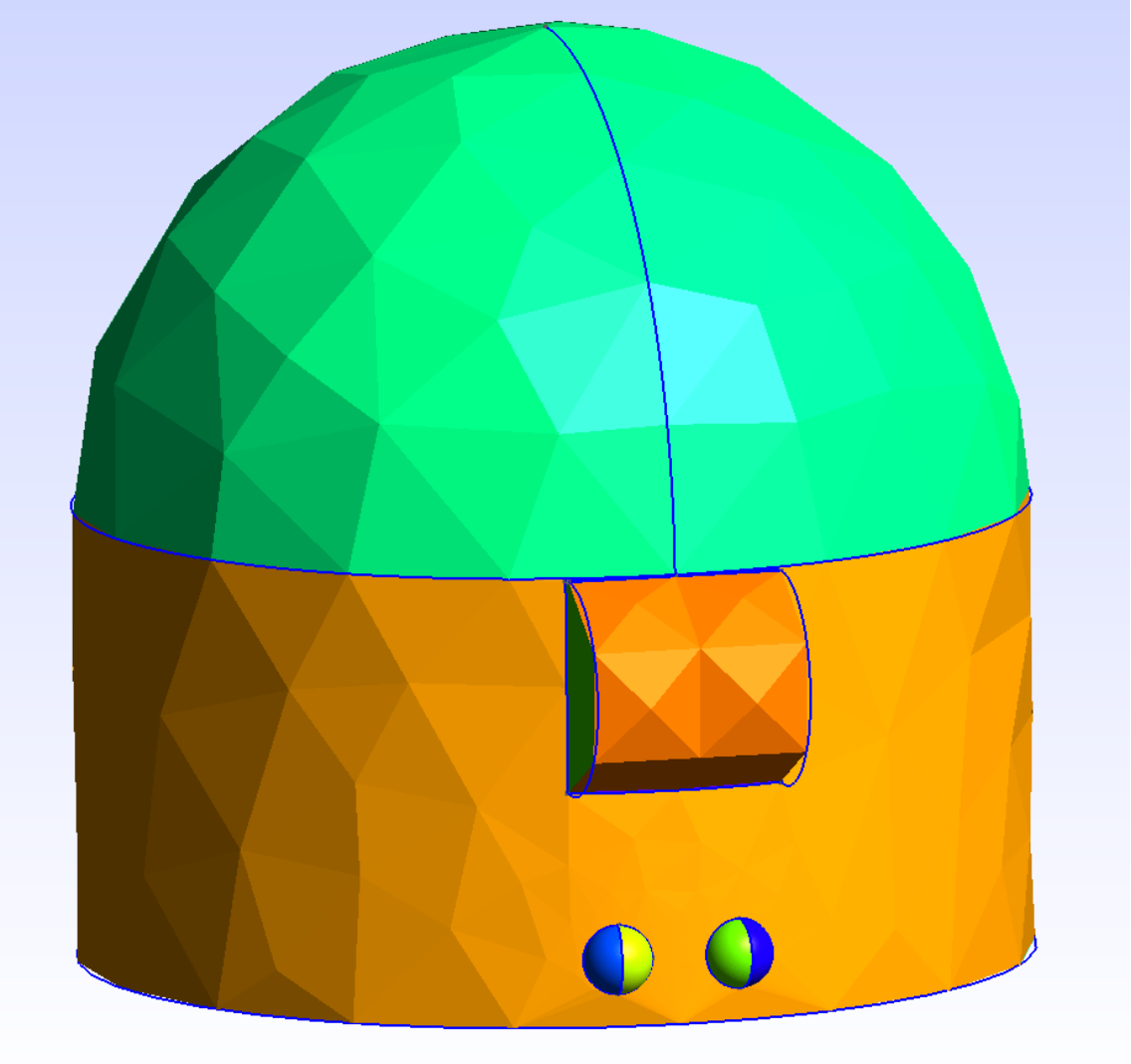


Figura 4.9: Imagen de Gmsh con el modelo del submarino y sensores según la configuración mas sensible obtenida.

5

Conclusiones

El resultado principal de este trabajo de fin de grado ha sido la implementación de un sistema paramétrico para simular campos eléctricos en el estudio de la electrorrecepción [2]. El sistema permite optimizar la posición y tamaño de los electrodos de un submarino robot, enfocado a la detección de obstáculos de tipo pared.

Recibiendo un fichero de configuración inicial, nuestro software de control es capaz de generar paramétricamente la geometría CAD, variando automáticamente los parámetros requeridos. A partir de esta geometría se crea una malla usando el software `Gmsh` y se ejecuta una simulación usando la malla especificada junto a las condiciones de contorno dictadas por el fichero de configuración inicial. Finalmente nuestro software recoge y prepara los resultados para su post-procesado.

La virtualización implementada nos ha permitido continuar esta investigación incluso tras la situación de trabajo remoto creada por el COVID-19, y además ha permitido automatizar el proceso de optimización y búsqueda de parámetros. Adicionalmente se ha implementado teniendo en cuenta la ejecución en entornos de cómputo variados (máquinas multinúcleo y MPI para el clúster de la EPS). Este sistema virtualizado, automático y paralelo acelera los experimentos, y permite definir con precisión las pruebas óptimas a implementar en el robot submarino real.

Trabajo futuro

Los resultados de este trabajo y el prometedor potencial de su implantación futura en un robot submarino real me han motivado para continuar con mis estudios de máster para desarrollar todas las capacidades de esta tecnología. Continuando la línea de investigación, implementaremos el sistema electro-receptor en el robot submarino para validar la distribución de sensores optimizada en un entorno real. Esto es, usando electrodos esféricos de acero inoxidable, una posible solución será desarrollar una estructura impresa en 3D (para su rápido prototipado) que permita el ajuste robusto de la posición de los electrodos. Inicialmente esto se validará en una piscina ya que es un entorno controlado y poco susceptible a factores externos. Todo con el objetivo final de dotar con este nuevo sentido eléctrico a los robots submarinos y facilitar su trabajo autónomo en los entornos acuáticos reales.

Bibliografía

[1] B. A. Carlson, J. A. Sisneros, A. N. Popper, and R. R. Fay, *Electroreception: Fundamental Insights from Comparative Approaches*. Springer, 2019.

[2] T. H. Bullock, C. D. Hopkins, A. N. Popper, and R. R. Fay, "Electroreception," 2005.

[3] I. D. Neveln, Y. Bai, J. B. Snyder, J. R. Solberg, O. M. Curet, K. M. Lynch, and M. A. MacIver, "Biomimetic and bio-inspired robotics in electric fish research," *Journal of experimental Biology*, vol. 216, no. 13, pp. 2501–2514, 2013.

[4] F. Boyer, P. B. Gossiaux, B. Jawad, V. Lebastard, and M. Porez, "Model for a sensor inspired by electric fish," *IEEE transactions on robotics*, vol. 28, no. 2, pp. 492–505, 2011.

[5] Y. Morel, V. Lebastard, and F. Boyer, "Neural-based underwater surface localization through electrolocation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2596–2603, IEEE, 2016.

[6] J. R. Solberg, K. M. Lynch, and M. A. MacIver, "Active electrolocation for underwater target localization," *The International Journal of Robotics Research*, vol. 27, no. 5, pp. 529–548, 2008.

[7] V. Lebastard, C. Chevallereau, A. Girin, F. Boyer, and P. B. Gossiaux, "Localization of small objects with electric sense based on kalman filter," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1137–1142, IEEE, 2012.

[8] V. Lebastard, C. Chevallereau, A. Girin, N. Servagent, P.-B. Gossiaux, and F. Boyer, "Environment reconstruction and navigation with electric sense based on a kalman filter," *The International Journal of Robotics Research*, vol. 32, no. 2, pp. 172–188, 2013.

[9] S. B. Ferrer, V. Lebastard, and F. Boyer, "Autonomous underwater docking using active/passive electro-location,"

[10] S. Mintchev, C. Stefanini, A. Girin, S. Marrazza, S. Orofino, V. Lebastard, L. Manfredi, P. Dario, and F. Boyer, "An underwater reconfigurable robot with bioinspired electric sense," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1149–1154, 2012.

[11] V. H. García García *et al.*, "Desarrollo de un prototipo de sistema electro-localizador en entornos acuáticos," Master's thesis, 2016.

[12] "Inspector 3 — seadrone." `https://seadronepro.com/inspector-3`. (Accessed on 07/14/2020).

[13] "Elmer models manual." `http://www.nic.funet.fi/index/elmer/doc/ElmerModelsManual.pdf`. (Accessed on 07/20/2020).

[14] D. Baguley, D. Hose, E. K. N. A. for Finite Element Methods, and Standards, *Why do-finite element analysis?* NAFEMS, 1994.

[15] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[16] C. Geuzaine and J.-F. Remacle, “Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities,” *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[17] M. Malinen, “Elmer finite element solver for multiphysics and multiscale problems,” 2013.

[18] J. Ahrens, B. Geveci, and C. Law, “Paraview: An end-user tool for large data visualization,” *Visualization Handbook*, 01 2005.

[19] V. Lebastard, F. Boyer, C. Chevallereau, and N. Servagent, “Underwater electro-navigation in the dark,” in *2012 IEEE International Conference on Robotics and Automation*, 2012 IEEE International Conference on Robotics and Automation, (Saint Paul, Minnesota, USA, United States), pp. 1155 – 1160, 2012.

[20] “pygmsh 4.3.6 documentation.” `https://pygmsh.readthedocs.io/en/latest/index.html`. (Accessed on 07/14/2020).

[21] W. J. Schroeder, K. Martin, and W. E. Lorensen, *The visualization toolkit*. [Clifton Park, NY]: Kitware, 4. ed. ed., 2006.

[22] “Mpi for python — mpi for python 3.0.3 documentation.” `https://mpi4py.readthedocs.io/en/stable/index.html`. (Accessed on 07/15/2020).

[23] M. P. Forum, “Mpi: A message-passing interface standard,” tech. rep., USA, 1994.

Apéndices

A

Repositorio de código

Este proyecto se ha realizado con herramientas libres y Open Source (FOSS), y todo el código desarrollado ha sido publicado también bajo licencias abiertas. Se encuentra disponible en el siguiente repositorio de GitHub:

`https://github.com/GNB-UAM/3D-electroreception-simulator`

Además del código se incluyen las configuraciones de todos los resultados presentados, para facilitar su replicación, así como los ficheros de control e instrucciones de instalación de las librerías y dependencias.

Con esta aportación esperamos fomentar la presencia de las plataformas libres en la investigación. En concreto queremos hacer más accesible el estudio de los sistemas electrorreceptivos, para su estudio en la naturaleza y para su aplicación en robótica submarina.

B

Ficheros de configuración

Código B.1: Ejemplo de fichero para generar la geometría del robot submarino usando Gmsh

```
1  // This code was created by pygmsh vunknown.
2  SetFactory("OpenCASCADE");
3  Mesh.CharacteristicLengthMin = 0.01;
4  Mesh.CharacteristicLengthMax = 0.03;
5  vol0 = newv;
6  Sphere(vol0) = {0, 0, 0, 0.155, 0};
7  vol1 = newv;
8  Cylinder(vol1) = {0, 0, 0, 0, 0, -0.147, 0.155};
9  vol2 = newv;
10 Sphere(vol2) = {0.1536739535129406, 0.020231559794107994, -0.125, 0.01125};
11 Rotate { {0, 0, 1}, {0.1536739535129406, 0.020231559794107994, -0.125}, 0.1308996938995747 }
       {Volume{vol2}; }
12 Rotate { {0, 0, 1}, {0, 0, 0}, 0.0 } {Volume{vol2}; }
13 vol3 = newv;
14 Sphere(vol3) = {0.1536739535129406, -0.020231559794107994, -0.125, 0.01125};
15 Rotate { {0, 0, 1}, {0.1536739535129406, -0.020231559794107994, -0.125}, -0.1308996938995747 }
       {Volume{vol3}; }
16 Rotate { {0, 0, 1}, {0, 0, 0}, 0.0 } {Volume{vol3}; }
17 Translate {0, 0, 0} { Volume{vol0}; }
18 Translate {0, 0, 0} { Volume{vol1}; }
19 Translate {0, 0, 0} { Volume{vol2}; }
20 Translate {0, 0, 0} { Volume{vol3}; }
21 Rotate { {0, 0, 1}, {0, 0, 0}, 0.0 } {Volume{vol0}; }
22 Rotate { {0, 0, 1}, {0, 0, 0}, 0.0 } {Volume{vol1}; }
23 Rotate { {0, 0, 1}, {0, 0, 0}, 0.0 } {Volume{vol2}; }
24 Rotate { {0, 0, 1}, {0, 0, 0}, 0.0 } {Volume{vol3}; }
25 vol4 = newv;
26 Box(vol4) = {-1.0, -1.0, -1.0, 2, 2, 2};
27 bo1[] = BooleanDifference{ Volume{vol4}; Delete; } {
       Volume{vol0};Volume{vol1};Volume{vol2};Volume{vol3}; Delete;};
28 Transfinite Line {50, 58, 49, 51, 46, 53, 47, 48} = 15 Using Progression 1;
```

Código B.2: Ejemplo de fichero para definir una simulación de campo eléctrico con Elmer

```
Header
  CHECK KEYWORDS Warn
  Mesh DB "." "."
  Include Path ""
  Results Directory ""
End

Simulation
  Max Output Level = 5
  Coordinate System = Cartesian
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = Scanning
  Steady State Max Iterations = 1
  Output Intervals = 1
  Timestepping Method = BDF
  BDF Order = 1
  Timestep intervals = 3
  Timestep Sizes = 1
  Solver Input File = case.sif
  Post File = case.vtu
End

Constants
  Gravity(4) = 0 -1 0 9.82
  Stefan Boltzmann = 5.67e-08
  Permittivity of Vacuum = 8.8542e-12
  Boltzmann Constant = 1.3807e-23
  Unit Charge = 1.602e-19
End

Body 1
  Target Bodies(1) = 37
  Name = "Body␣1"
  Equation = 1
  Material = 1
End

Solver 1
  Equation = Static Current Conduction
  Variable = Potential
  Procedure = "StatCurrentSolve" "StatCurrentSolver"
  Exec Solver = Always
  Stabilize = True
  Bubbles = False
  Lumped Mass Matrix = False
  Optimize Bandwidth = False
  Steady State Convergence Tolerance = 1.0e-5
  Nonlinear System Convergence Tolerance = 1.0e-7
  Nonlinear System Max Iterations = 20
  Nonlinear System Newton After Iterations = 3
```


UAM
UNIVERSIDAD AUTONOMA
DE MADRID